Table of Contents

Endeavour

Endeavour Configuration Details

A new SGI UV 2000 system, Endeavour, has been deployed to replace Columbia and to provide resources for applications that need access to large cache-coherent, global shared-memory capabilities in a single system image (SSI).

Processors, Memory, and Performance

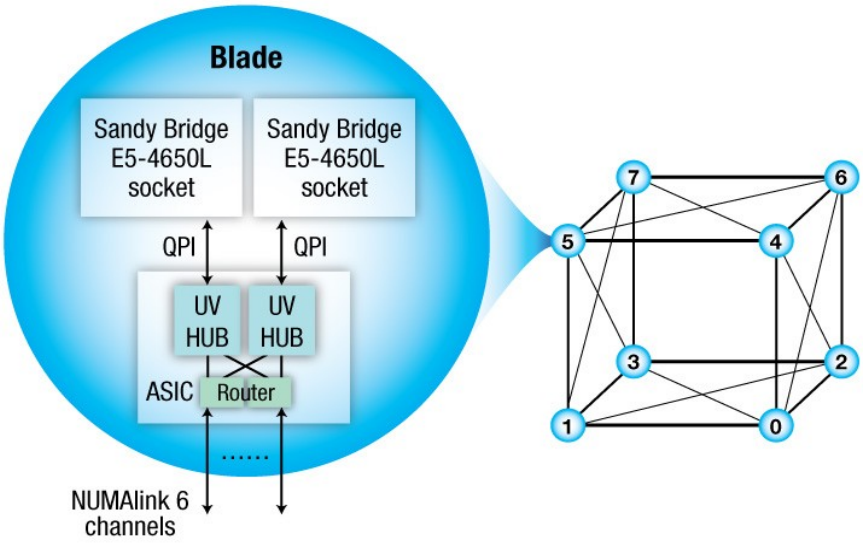
The Endeavour supercomputer comprises two nodes, Endeavour1 and Endeavour2. Endeavour1 includes 512 cores and 2 TB of memory and has a theoretical peak performance (double-precision floating point format) of 11 teraflops. Endeavour2, with 1,024 cores and 4 TB of memory, is double the size of Endeavour1 and has a peak performance of 21 teraflops. Together, these two nodes provide 32 teraflops, which is slightly higher than the 30 teraflops provided by the four Columbia nodes (c21-24) combined.

At a more detailed level, Endeavour is based on Intel Xeon E5-4650L (Sandy Bridge) 2.6-GHz processors. Each Endeavour blade has two sockets, and each socket contains a processor with 8 cores and access to four 8-GB memory DIMMs. This provides an average of 4 GB/core, which is double the size of the memory per core in Columbia and in the E5-2670 Sandy Bridge compute nodes in Pleiades.

Except for the memory size, the configuration inside each Endeavour E5-4650L socket is the same as that in the E5-2670 sockets in Pleiades. However, unlike Pleiades, there are no direct Intel Quick Path Interconnect (QPI) links connecting the two E5-4650L sockets on an Endeavour blade.

System-wide Cache-coherent Interconnect

The most important feature of the UV 2000 architecture is its system-wide cache-coherent interconnect. The following diagram shows the interconnect at the blade level (left) and at the individual rack unit (IRU) level (right).



The diagram shows the connections, as follows:

- At the blade level, each socket is connected via a cache-coherent QPI link to a UV hub inside an application-specific integrated circuit (ASIC). Multiple NUMAlink channels provide the connections out of the blade.
- At the IRU level, the ASICs of the eight blades within each IRU are connected through the NUMAlink channels to form a 3D enhanced hypercube--with a maximum of two hops between any two blades--providing cache-coherency within the IRU. The IRUs are connected to each other through cache-coherent NUMAlink 6 routers, providing cache-coherency throughout the system. So, the UV 2000 nodes provide cache-coherent non-uniform memory access (ccNUMA).

The latency for a process to access data in memory varies depending on the distance between the socket/blade where the process is running, and the socket/blade that contains the data.

In addition to cache coherency, the UV hub provides other functions to enable efficient system operation. See the SGI white paper Technical Advances in the SGI UV Architecture (PDF, 546 KB) for more information.

To get started, see Preparing to Run on Endeavour.

Processor, Memory, and Network Subsystems Statistics

Detailed configuration statistics for Endeavour are listed below:

Endeavour Processor and Memory Subsystems Statistics

	Endeavour	
	Endeavour1	**Endeavour2**
Architecture	UV 2000	
Processors	8-core Xeon E5-4650L	
Hyper-Threading	OFF	
Turbo Boost	ON	
Maximum Double-Precision Flops per Cycle per Core	8	
# of Cores/Socket	8	
# of Sockets/Blade	2	
Total # of Blades	32	64
Total # of Cores	512	1024
Total Double Precision Tflops	11	21
L1 Instruction and Data Caches	32KB/Core Cache line size: 64B Associativity: 8	
L2 Cache	256 KB/Core Cache line size: 64B Associativity: 8	
L3 Cache	20 MB shared by 8 cores Cache line size: 64B Associativity: 20	
Memory/Core	4 GB	
Memory Speed	1600 MHz	
Total Memory/Socket	32 GB	
Total Memory	2 TB	4 TB
Interconnect	NUMAlink 6	
Topology	5D enhanced hypercube	Crossbar-connected 3D enhanced hypercube
Home Filesystem	Pleiades /u/username	
Lustre Filesystem	Pleiades /nobackup/username	

Preparing to Run on Endeavour

Overview of Endeavour

The Endeavour supercomputer comprises two nodes, Endeavour1 and Endeavour2, and has been installed to replace Columbia as the new shared memory system at NAS. Endeavour1 includes a total of 512 cores and 2 TB of global shared memory. Endeavour2 is double the size of Endeavour1, with 1024 cores and 4 TB of global shared memory. See Endeavour Configuration Details for more hardware information.

Endeavour is running with the SUSE Linux Enterprise Server 11 Service Pack 2 (SLES 11 SP2) operating system.

Endeavour uses Pleiades front ends (PFEs), bridge nodes, and filesystems, and shares some of the Pleiades InfiniBand fabric. However, Endeavour uses its own designated PBS server.

Connecting to Endeavour

To access Endeavour, submit PBS jobs from the Pleiades front end systems (PFEs) or bridge nodes. You cannot log directly into Endeavour from the PFEs or bridge nodes unless you have a PBS job running there.

To use Endeavour, you need a Pleiades account and an Endeavour SBU allocation. After you acquire your Endeavour SBU allocation, the access control lists (ACLs) of the appropriate Endeavour PBS queues will be updated to include your project IDs (GIDs).

Accessing your Data

Your Pleiades home filesystem (/u/*username*) is your home filesystem for both Pleiades and Endeavour. The same applies to your Pleiades Lustre Filesystem (/nobackup). Please note that the Columbia CXFS filesystem is not mounted on Endeavour.

NOTE: The default stripe count on your Pleiades Lustre /nobackup filesystem is 1. Depending on the I/O pattern of your application, you may want to adjust the stripe count to a larger size for directories or files to get better performance, as shown in the following example:

```
pfe20% cd /nobackup/username
pfe20% mkdir bigstripe_dir
pfe20% lfs setstripe -c 32 bigstripe_dir
```

For more information on using Lustre filesystems, see Lustre Basics and Lustre Best Practices.

If you need to migrate data from your Columbia home filesystem or CXFS /nobackup filesystem to your Pleiades home filesystem or Lustre /nobackup filesystem, use remote file transfer commands such as shiftc, bbftp, or **scp**. The **shiftc** command is highly recommended for its ease of use and performance, and is shown in the following example:

```
pfe20% cd /nobackup/username
pfe20% shiftc -rp cfe2:/nobackup2a/username/dir1 ./bigstripe_dir
```

Compiling and Running your Code

Because the Columbia (IA-64) and Endeavour (x86-64) processor architectures are different, an executable that was compiled on Columbia cannot be used on Endeavour. To use executables that were built on Columbia, you must copy your source code from Columbia to your Pleiades home filesystem or Lustre /nobackup filesystem, and recompile the code on the Pleiades front-end systems.

Unlike on Columbia, there are no default modules for compilers, Message Passing Interface (MPI) components, or math libraries loaded on Pleiades. To use these modules, you must first load them. The following sections list the recommended modules for building and running your applications.

OpenMP Applications

Load the latest Intel compiler module and use the Intel compiler **-openmp** option to build the executable, as follows:

```
pfe20% module load comp-intel/2012.0.032
pfe20% ifort -O2 -openmp program.f
```

Set a specific number of OpenMP threads in your PBS job, and use a thread-pinning method to ensure that the threads do not get in the way of each other on the same core or migrate from one core to another. For example:

```
#PBS ...

module load comp-intel/2012.0.032
setenv OMP_NUM_THREADS 16

setenv KMP_AFFINITY compact # or: setenv KMP_AFFINITY scatter
./a.out > output

or

setenv KMP_AFFINITY disabled
```

```
dplace -x2 ./a.out > output
```

See Process/Thread Pinning Overview for more information about different pinning methods.

Applications that Require Scientific and Math Libraries

SGI's Scientific Computing Software Library (SCSL) library is not available on Pleiades or Endeavour. If you used an application on Columbia that called a routine in the SCSL library, you must change your application to call a corresponding Intel Math Kernel Library (MKL) routine instead.

TIP: After you load an Intel compiler module (versions 11.1 and later), you do not need to load an MKL module. It is included in the Intel compiler module.
See MKL to learn how to link to proper MKL libraries. In many cases, the following commands may be sufficient:

```
pfe20% module load comp-intel/2012.0.032
pfe20% ifort -O2 program.f -mkl
```

NOTE: By itself, the **-mkl** option implies **-mkl=parallel**, which will link to the threaded MKL library. If your application can benefit from using multiple OpenMP threads when the MKL routines are called, you should also set the environment variable **OMP_NUM_THREADS**, as shown in the OpenMP example in the previous section.

If you want to use a non-threaded MKL library, change the **-mkl** option to **-mkl=sequential**.

MPI Applications

SGI's latest Message Passing Toolkit (MPT) library is recommended. Load the library as shown in the following example:

```
pfe20% module load comp-intel/2012.0.032
pfe20% module load mpi-sgi/mpt.2.06rp16
pfe20% ifort -O2 program.f -lmpi
```

During runtime, load the Intel compiler and the MPT modules and use **mpiexec** to launch the executable:

```
#PBS ...
module load comp-intel/2012.0.032
module load mpi-sgi/mpt.2.06rp16

mpiexec -np xx ./a.out > output
```

Running PBS jobs

Although Endeavour uses Pleiades front ends (PFEs), bridge nodes, and filesystems, and shares the Pleiades InfiniBand fabric for I/O, it uses a separate PBS server: pbspl3. So, PBS jobs cannot run across the Endeavour and Pleiades compute nodes.

Job Accounting

The minimum allocatable unit (MAU) of Endeavour is 8 cores with 30 GB of memory. Using 1 MAU per hour, the SBU rate on Endeavour is 0.74.

To find the status of the Endeavour allocation for your group, such as the numbers of SBUs used and remaining, use the following command:

```
pfe20% acct_ytd -c endeavour your_GID
```

You can also view the SBU usage for each job. For example, to view the statistics of all of your jobs that ran on Endeavour2 on February 22, 2013, type:

```
pfe20% acct_query -c endeavour2 -d 02/22/13 -u your_username -olow
```

See Job Accounting Utilities to learn more about using the *acct_ytd* and *acct_query* tools.

Endeavour PBS Queues

Some Endeavour resources are set aside for system operations, as follows:

- **Endeavour1:** 8 cores are set aside; 504 cores with 1,890 GB are the maximum resources available for PBS jobs
- **Endeavour2:** 16 cores are set aside; 1,008 cores with 3,780 GB are the maximum resources available for PBS jobs

Endeavour queue names have an "e_" prefix. Among them, the higher-priority e_debug queue is the only one that has a limit (128 cores per job). If you want to use the e_debug queue, you must specify **`-q e_debug`** in either the qsub command line or inside the PBS script.

The following Endeavour PBS queues are available for general use:

```
pfe20% qstat -Q @pbspl3

Queue     Ncpus/      Time/         State counts
name        max/def    max/def   jm T/Q/H/W/R/E/B pr
========= =====/=== ======/===== == ============= ===
```

```
e_normal    --/  8  08:00/01:00 -- 0/4/0/0/0/0/0  0
e_long      --/  8  72:00/36:00 -- 0/0/0/0/0/0/0  0
e_vlong     --/  8 600:00/24:00 -- 0/0/0/0/0/0/0  0
e_debug    128/  8  02:00/00:30 -- 0/0/0/0/0/0/0 15
```

PBS Commands

Run PBS commands (such as **qsub**, **qstat**, and **qdel**, etc.) from the PFEs or bridge nodes and specify the Endeavour PBS server, pbspl3. For example:

```
pfe20% qsub -q queue_name@pbspl3 job_script
pfe20% qstat -nu username @pbspl3
pfe20% qstat jobid.pbspl3
pfe20% qdel jobid.pbspl3
```

WARNING: When you run the qsub command from the PFEs or bridge nodes, you must specify **@pbspl3** or an Endeavour queue name. Otherwise, the job will be routed to run on Pleiades. (When you run the qstat command, you must specify **@pbspl3**.)

If you do not normally run jobs on Pleiades compute nodes, and your primary workload is on Endeavour, you may want to consider setting the PBS_DEFAULT environment variable to pbspl3. This allows you to simplify the PBS commands, as follows:

```
pfe20% setenv PBS_DEFAULT pbspl3
pfe20% qsub -q job_script
pfe20% qstat -nu username
pfe20% qstat jobid
pfe20% qdel jobid
```

Job Submission Examples

The following examples demonstrate how to request interactive PBS sessions to run on Endeavour. You can also include similar resource requests in PBS scripts for batch jobs.

- To request 48 cores:

  ```
  pfe20%  qsub -I -lncpus=48 -q queue_name@pbspl3
  ```

  The resources allocated to this job will be 6 MAUs (48 cores and 180 GB).
- To request 128 GB of memory:

  ```
  pfe20%  qsub -I -lmem=128GB -q queue_name@pbspl3
  ```

  To satisfy the request for 128 GB, the resources allocated will be 5 MAUs (40 cores and 150 GB).

- To request 8 cores and 128 GB of memory, use one of the following:

```
pfe20%  qsub -I -lncpus=8,mem=128GB -q queue_name@pbspl3
pfe20%  qsub -I -lselect=ncpus=8:mem=128GB -q queue_name@pbspl3
```

To satisfy both the ncpus and mem requests, the resources allocated will be 5 MAUs (40 cores and 150 GB).

- If you do not specify which Endeavour node to run your job, PBS will assign one for you. The following example specifies Endeavour1 and requests 48 cores:

```
pfe20%  qsub -I -lselect=host=endeavour1:ncpus=48 -q queue_name@pbspl3
```

If your job begins to use more memory than allocated, it will be killed by a policykill daemon and an email will be sent to you.